2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

# Cross-Agent Relocalization for Decentralized Collaborative SLAM

Philipp Bänninger, Ignacio Alzugaray, Marco Karrer and Margarita Chli

***Abstract*— State-of-the-art decentralized collaborative Simultaneous Localization And Mapping (SLAM) systems crucially lack the ability to effectively use well-mapped areas generated by other agents in the team for relocalization. This often leads to map redundancy between agents, inefficient communication, and the need for costly re-mapping of areas previously mapped by other agents. In this work, we propose a strategy to efficiently share the areas mapped by different agents in a collaborative, decentralized SLAM system. This approach directly addresses map redundancy while maintaining the consistency of the estimates across the agents and keeping the overall system scalable in terms of cross-agent communication and individual computational effort. Our method leverages covisibility information between keyframes instantiated by different agents to transfer local sub-maps on-the-fly in a completely decentralized, peer-to-peer fashion. A globally consistent estimate is achieved by solving a distributed bundle adjustment problem using the Alternating Direction Method of Multipliers (ADMM), where we enforce constraints on shared map points and keyframes across agents.**

## I. Introduction

Considering the maturity of single-agent SLAM, research efforts have recently shifted towards investigating the challenges of collaborative SLAM using a team of agents. Many real-world tasks can be achieved significantly more efficiently by distributing the workload among multiple agents, rendering the importance of multi-agent collaboration and, in particular, collaborative SLAM seminal, with potential applications such as shared AR experiences, using smartphones [1] or search-and-rescue scenarios using a team of robots [2]. Due to their ubiquity, cost-effectiveness, and ability to provide information-rich output, cameras have traditionally been a suitable choice for both single-agent [3]–[5] as well as multi-agent SLAM [2], [6], [7], often considering parallel tracking, mapping and loop closure modules. In particular, collaborative SLAM approaches can be designed as centralized or decentralized systems. In centralized architectures [2], [7], [8], local, per-agent tracking and mapping data is communicated to a central server. The server then establishes intra- and inter-agent constraints and periodically optimizes the global solution (e.g., map, motion), potentially communicating back to the individual agents to improve their onboard estimate. The computational and communication requirements on the server in centralized approaches scale, at best, linearly with the number of agents, becoming a prohibitive bottleneck for large, swarm-sized teams of robots [9]. Alternatively, in decentralized collaborative architectures the computational efforts are distributed among all the robots and peer-to-peer communication is possible. Nonetheless, decentralized architectures must rely on efficient and reliable coordination strategies to establish inter-agent constraints and keep consistent state estimates across the team. Previous research efforts have focused on peer-to-peer, communication-efficient solutions for decentralized place recognition [10], [11] and loop closure [12], [13], as well as distributed optimization approaches [14]–[16].

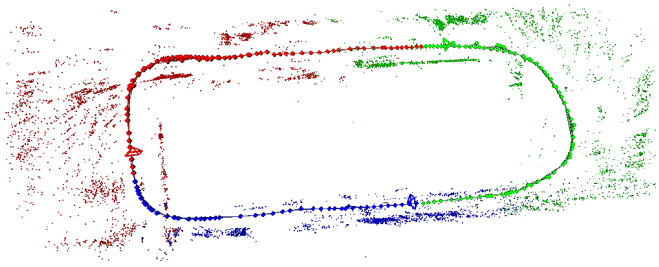


Fig. 1: Example of a real office scene where three agents (in different colors) perform decentralized collaborative SLAM. Using the proposed approach, agents are able to re-use and relocalize themselves in each other's map by distributing the SLAM data on-demand among agents.

The ability to relocalize in previously mapped areas of the environment has become an established practice in state-of-the-art single-agent SLAM [4], [5], and has also made its way in centralized collaborative SLAM [7], as mapping efforts and drift can be dramatically reduced in previously explored areas. However, in order to leverage and relocalize in other agents' maps, agents in state-of-the-art decentralized SLAM systems often create, maintain and transmit redundant information on previously mapped areas. Inspired by the single agent approach [17], here we propose an approach to leverage the covisibility information and efficiently identify and share relevant parts of the map with other agents in the team to be reused by them (see Fig. 1). To yield a consistent shared map estimate across agents in a decentralized and asynchronous manner we rely on the distributed ADMM bundle adjustment formulation [15].

In brief, this paper presents the following contributions:

1) an open-source[1] decentralized map management strategy for efficient cross-agent relocalization that mitigates map redundancy,
2) a strategy to leverage asynchronous ADMM to maintain global consistency over estimates of shared states, and
3) thorough evaluations on state-of-the-art and self-collected datasets revealing the scalability of our ap-

P. Bänninger, M. Karrer and M. Chli are with the Vision for Robotics Lab, ETH Zürich, Switzerland: {baephili, karrerm, chlim}@ethz.ch, and I. Alzugaray is with the Department of Computing, Imperial College London, UK: i.alzugaray@imperial.ac.uk

[1] https://github.com/VIS4ROB-lab/decoSLAM.

proach in challenging multi-agent scenarios, with direct impact on the communication overhead.

## II. Related Work

Centralized approaches to vision-based collaborative localization and mapping [2], [7], [8] rely on computationally powerful servers to relieve the individual agents of costly processes, such as global inference and global relocalization. Conversely, decentralized approaches have to rely on scalable peer-to-peer interactions to overcome the need for a centralized server.

Early works in decentralized SLAM used peer-to-peer collaboration to exchange local information in the form of condensed factor graphs among close-by robots, for both localization [18] and SLAM problems [19]. In the same vein, Dubois et al. [20] consider visual-inertial factor graphs and make sparsified factors available to agents within communication range. Murai et al. [21] show scalable decentralized multi-robot localization using Gaussian belief propagation.

Focusing specifically on pose-to-pose constraints, Tian et al. [14] explore a robust, distributed and asynchronous Pose Graph Optimization (PGO), and further provide certifiable correctness in [22]. Similar techniques are expanded into complete multi-agent systems in [23]–[25].

In visual place recognition, different techniques often rely on lossy compression of images (e.g. descriptors) to effectively characterize and efficiently re-identify unique places. For single agent systems, bag-of-words-based [26] approaches are a common choice [4], [27] to achieve such a representation. Conversely, full-image descriptors were shown to perform well in decentralized candidate selection [10], [11] for multi-agent SLAM. Taking this a step further, [12] and [13] address the communication efficiency of the subsequent geometric verification step for better loop closures. Furthermore, the work of Choudhary et al. [16] combines object-based maps with a PGO backend to reduce the communication burden when establishing inter-agent constraints.

A recent comparative study on multi-agent optimization by Halsted et al. [28] suggests the viability of ADMM as a powerful choice for distributed optimization in multi-robot systems. The use of ADMM as a decentralized optimization backbone for multi-agent problems was first investigated in [29], for pose-graph optimization, and in [30], [31] for offline large-scale bundle-adjustment. Theoretical contributions on asynchronous ADMM [15] have recently been explored in collaborative state estimation for two agents with communication delays [32] and scaling up constellation-estimation to a swarm of aerial robots [9].

The literature so far has focused mainly on minimizing the communication efforts needed to obtain a collaborative trajectory estimate across the agents, generally achieved using a distributed pose graph optimization with inter-agent constraints. These schemes, however, often disregard the re-use of maps collected from other agents to directly improve the local, per-agent estimates (e.g., pose tracking, relocalization), limiting the benefit of the collaboration to the optimization of the joint trajectory. Here, we purposefully enforce the reuse of the acquired and processed data across all agents in all steps of the system, including the frame-to-frame estimation and thus also directly influence the mapping process.

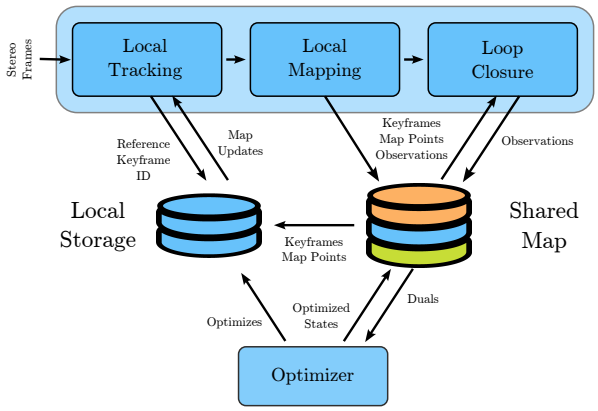



Fig. 2: Each agent runs a standard SLAM pipeline that stores and loads keyframes and map points from an abstract shared map. Locally stored data is used during local tracking and optimized in a distributed global optimization.

## III. Methodology

The diagram in Fig. 2 depicts the system components present on each agent and the interactions between them. Each agent runs a SLAM system (Section III-A) that creates new keyframes, map points, and associations between them (i.e. observations), which are stored in a distributed shared map (Section III-B). Every agent caches a minimal amount of information in their local storage to minimize data duplicity and the communication overhead required to keep it consistent across agents. Global consistency of map points and keyframes is achieved using a distributed global optimization scheme (Section III-C).

In the following, we denote the set of agents in the decentralized system as $\mathcal{R} := \{1, 2, \ldots, R\}$, and the sets of map points and keyframes stored in the shared map as $\mathcal{P}$ and $\mathcal{T}$, respectively.

### A. SLAM pipeline

Each agent runs a custom-made graph-based, stereo SLAM onboard using standard modules [17], [27].

*1) Local Tracking:* We localize new stereo frames using 2D-to-3D matching using a RANSAC-based P3P [33]. Each incoming frame is assigned to a reference keyframe $T_{\text{ref}}$ with the strongest covisibility, i.e. with the highest number of commonly observed map points. At the same time, as illustrated in Fig. 3, we define a subset of local keyframes $\mathcal{T}_{\text{local}}(T_{\text{ref}}) \subseteq \mathcal{T}$ that includes the reference keyframe $T_{\text{ref}}$ and its neighbors in the covisibility graph. The tracking of each new frame only considers the map points observed from the subset of local keyframes $\mathcal{T}_{\text{local}}(T_{\text{ref}})$ [17], [34]. As tracking proceeds, the reference keyframe $T_{\text{ref}}$ might eventually switch to one of the neighboring keyframes denoted $T'_{\text{ref}} \in \mathcal{T}_{\text{local}}(T_{\text{ref}})$. As a consequence, the new reference

keyframe and covisible neighboring keyframes $\mathcal{T}_{\text{local}}(T'_{\text{ref}})$, and their associated map points are dynamically fetched from the shared map into local storage (Section III-B), potentially re-using information created by other agents. A new keyframe is created when the ratio between the average depth of the observed map points and the baseline to the reference keyframe exceeds a threshold or the number of observed points falls below 40, as in [3], [5].

*2) Local Mapping:* Unmatched keypoints in new keyframes are stereo-triangulated into new map points and submitted to the shared map. These new map points are also projected into the keyframe's neighbors in the covisibility graph to introduce additional observations. As in [5], map points are generously created but aggressively culled, removing them if they have not been re-observed after 5 keyframes since their creation.

*3) Loop Closure:* For each new keyframe a set of loop-closure candidates are identified in a decentralized manner as described in [11], employing NetVLAD descriptors [35], and ignoring those neighboring candidates which are connected to the new keyframe in the covisibility graph. Geometric verification of loop-closure candidates is applied by matching the keypoints of the candidate with the map points observed in the new keyframe, applying 2D-3D RANSAC and only accepting those candidates with a minimum number of inliers. After a successful loop closure, duplicated map points between the new keyframe and the candidate keyframe are merged.

### B. Distributed Shared Map

Keyframes, map points, and observations are handled in the SLAM process as if each agent would be able to seamlessly interact with a global, shared map. In our decentralized approach, this is only an abstraction as the data is not centrally stored in any specific agent, but distributed among all of them instead. The interactions with this shared map may then incur inter-agent communication overhead and thus, we propose a strategy efficiently store and share data on-demand, guided by the SLAM process.

*1) Distributed Look-up:* Inspired by [11], each newly created keyframe and map point is linked to a unique key represented by a tuple $(r, \text{ID})$, i.e. the agent's index $r \in \mathcal{R}$ and a unique ID assigned to that keyframe or map point upon creation by agent $r$, forming a so-called keyframe key or map point key, respectively. Each agent is designated as the responsible coordinator of the keyframes and map points they create. New observations related to specific keyframes and map points are transmitted to the appropriate coordinator. Any distributed look-up comprises a request to the relevant coordinator agent using a querying key, which, in response, returns the requested data.

*2) Covisibility Graph:* For efficient querying of the relevant map data, a covisibility graph consisting of vertices (i.e. keyframe keys) connected through weighted edges (indicating the number of co-observed map points) is maintained. Each agent stores vertices of the global covisibility graph for the keyframes that it created along with the corresponding

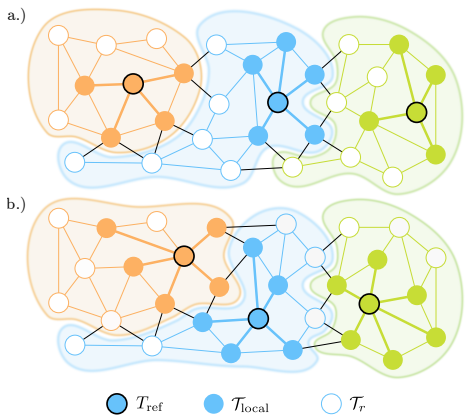



Fig. 3: On-demand map transfer based on covisibility graph between 3 agents, where each agent holds a set of keyframes $\mathcal{T}_r$ in local storage. a.) Agents consider a subset of locally stored keyframes $\mathcal{T}_{\text{local}}(T_{\text{ref}})$ around a reference $T_{\text{ref}}$ to identify map points for tracking. b.) A change of reference keyframe $T_{\text{ref}} \rightarrow T'_{\text{ref}}$ entails allocation of a new neighborhood $\mathcal{T}_{\text{local}}(T'_{\text{ref}})$, potentially transferring keyframes and the associated map points between agents.

edges, i.e. the keys of the connected neighboring keyframes. Following the distributed look-up scheme, each vertex and its edges can be accessed using the corresponding keyframe key. Edge weights are updated each time an observation is added between a keyframe $T_j$ and a map point $p_i$. The agent that coordinates the map point $p_i$ identifies all keyframes $\mathcal{T}(p_i) \subseteq \mathcal{T}$ observing $p_i$ and communicates the update to all agents coordinating a keyframe from the set $\mathcal{T}(p_i)$.

*3) Management of Local Storage:* Multiple agents concurrently requiring the same data in their SLAM (e.g. two agents tracking simultaneously the same map points), create duplicated copies in their local storage. The subsets of the global keyframes and map points locally stored in agent $r$ are referred to as $\mathcal{T}_r \subseteq \mathcal{T}$ and $\mathcal{P}_r \subseteq \mathcal{P}$, respectively. Locally stored copies of keyframe poses and map points positions are to be kept consistent across agents (Section III-C), and the coordinating agent of such data keeps a register of the agents that hold those copies (Section III-B.1). Specifically, every coordinator of a keyframe pose $T_j \in \mathcal{T}$ or map point position $p_i \in \mathcal{P}$, tracks all the agents that locally store copies of the same data, denoted $\mathcal{N}(T_j) \subseteq \mathcal{R}$ and $\mathcal{N}(p_i) \subseteq \mathcal{R}$.

Locally stored data is kept to a minimum in each agent to mitigate redundancy, only retaining those that are required by the local tracking module to operate (Section III-A.1). However, keyframes $T_j \notin \mathcal{T}_{\text{local}}(T_{\text{ref}})$ that are no longer in the neighborhood of the reference keyframe are further retained in local storage if no other agent has a copy of it according to the appropriate coordinator. Conversely, map points are retained as long as at least one keyframe exists in local storage that observes them, entailing that map points are only shared across inter-agent covisibility edges. Employing this strategy, each agent can request on-demand map transfers (i.e. map points and keyframes) from other agents via an intermediate known coordinator as they explore the scene

(Fig. 3), enabling cross-agent relocalization.

### C. Global Optimization

Keyframe poses and map point positions are optimized in a Global Bundle Adjustment (GBA) using a distributed version of ADMM [36], which we briefly introduce in the following lines.

*1) Distributed Asynchronous Bundle Adjustment:* Introducing equality constraints on duplicated keyframes and map points across multiple agents, the ADMM formulation by [15] allows to perform asynchronous and decentralized GBA using cross-agent consensus. Formally,

$$\forall p_{i,r} \in \tilde{\mathcal{P}}_r : \quad p_{i,r} = p_{i,s}, \qquad \forall s \in \mathcal{N}(p_i) \setminus r, \tag{1}$$

$$\forall T_{j,r} \in \tilde{\mathcal{T}}_r : \quad T_{j,r} = T_{j,s}, \qquad \forall s \in \mathcal{N}(T_j) \setminus r, \tag{2}$$

where $\tilde{\mathcal{T}}_r \subseteq \mathcal{T}_r$ and $\tilde{\mathcal{P}}_r \subseteq \mathcal{P}_r$ denote the subset of duplicate states and local copies stored on agent $r$ are denoted as $p_{i,r} \in \mathcal{P}_r$ and $T_{j,r} \in \mathcal{T}_r$, respectively. In [15], for every pairwise equality constraint, each involved agent introduces a dual variable. For instance, for a map point $p_i$ used within the subset of agents $\mathcal{N}(p_i)$, $(|\mathcal{N}(p_i)|-1)^2$ dual variables are introduced, where $|\mathcal{N}(p_i)|$ is the size of set $\mathcal{N}(p_i)$. The dual variable introduced on agent $r$ used to enforce the equality constraint on map point $p_i$ between agents $r$ and $s$, is denoted as $z_{i,r\to s}$ and, equivalently, a keyframe dual is denoted as $\lambda_{j,r\to s}$. The average dual for a map point $\bar{z}_{i,r}$ or keyframe pose $\bar{\lambda}_{j,r}$ are computed as

$$\bar{z}_{i,r} := \frac{1}{|\mathcal{N}(p_i)|-1} \sum_{s\in\mathcal{N}(p_i)\setminus r} z_{i,s\to r}, \tag{3}$$

$$\bar{\lambda}_{j,r} := \frac{1}{|\mathcal{N}(T_j)|-1} \sum_{s\in\mathcal{N}(T_j)\setminus r} \lambda_{j,s\to r}. \tag{4}$$

Note that computing the averages $\bar{z}_{i,r}$ and $\bar{\lambda}_{j,r}$ requires the communication of the per-agent duals to agent $r$. All average duals corresponding to map points $\bar{z}_{i,r}$ and keyframes $\bar{\lambda}_{j,r}$ that involve robot $r$ are referred to as the sets $\tilde{\Lambda}_r$ and $\tilde{\mathcal{Z}}_r$.

The ADMM algorithm iteratively and asynchronously alternates between two steps. In one step, the following augmented bundle adjustment problem is locally solved,

$$(\mathcal{T}_r^*, \mathcal{P}_r^*) = \underset{\mathcal{T}_r,\mathcal{P}_r}{\operatorname{argmin}} \, \mathbf{f}_{\mathrm{BA}}(\mathcal{T}_r, \mathcal{P}_r) + \mathbf{f}_{\mathrm{c}}\left(\tilde{\mathcal{T}}_r, \tilde{\Lambda}_r\right) + \mathbf{f}_{\mathrm{c}}\left(\tilde{\mathcal{P}}_r, \tilde{\mathcal{Z}}_r\right), \tag{5}$$

where $\mathbf{f}_{\mathrm{BA}}$ is a standard robust bundle adjustment cost function, consisting of reprojection errors (see e.g. [37]), and $\mathbf{f}_{\mathrm{c}}$ are the consensus terms enforcing the equality constraints in Eq. (1), which take the following form

$$\mathbf{f}_{\mathrm{c}}\left(\tilde{\mathcal{T}}_r, \tilde{\Lambda}_r\right) = \sum_j (\delta T_{j,r})^{\mathrm{T}} \bar{\lambda}_{j,r} + \frac{\gamma}{2}(|\mathcal{N}(T_j)|-1)\,\|\delta T_{j,r}\|^2 \tag{6}$$

$$\mathbf{f}_{\mathrm{c}}\left(\tilde{\mathcal{P}}_r, \tilde{\mathcal{Z}}_r\right) = \sum_i (p_{i,r})^{\mathrm{T}} \bar{z}_{i,r} + \frac{\gamma}{2}(|\mathcal{N}(p_i)|-1)\,\|p_{i,r}\|^2, \tag{7}$$

where $\gamma$ is a tuning parameter. As in [32], we use a reference pose $T_{j,\mathrm{ref}}$, which is the initial pose of the corresponding keyframe upon insertion into the map, and use the delta transformation expressed in the tangent space of this reference pose $\delta T_{j,r} = \log(T_{j,\mathrm{ref}}^{-1} T_{j,r})$. After solving Eq. (5), observations with residuals above a threshold are identified as outliers and removed.

In the other step, ADMM subsequently updates the duals

$$\lambda_{j,r\to s} = \lambda_{j,r\to s} - \tfrac{1}{2}\eta\,(\lambda_{j,r\to s} + \lambda_{j,s\to r}) - \eta\gamma\delta T_{j,r}^*, \quad \forall T_{j,r}^* \in \tilde{\mathcal{T}}_r^*, \forall s \in \mathcal{N}(T_j)\setminus r, \tag{8}$$

$$z_{i,r\to s} = z_{i,r\to s} - \tfrac{1}{2}\eta\,(z_{i,r\to s} + z_{i,s\to r}) - \eta\gamma p_{i,r}^*, \quad \forall p_{i,r}^* \in \tilde{\mathcal{P}}_r^*, \forall s \in \mathcal{N}(p_i)\setminus r, \tag{9}$$

where $\eta$ is a tunable stepsize parameter of the update, generally set within the range $(0, 1]$.

*2) Dual Update:* Other approaches using asynchronous ADMM [9], [32] commonly execute the aforementioned steps by broadcasting the dual updates $\lambda_{j,r\to s}$ or $z_{i,r\to s}$ directly to all other agents holding duplicates of that data. In our system, optimized duplicated variables by each agent (Eq. (5)) are sent to the appropriate coordinator agents of such variables. In turn, each contacted coordinator computes the dual updates (Eq. (8) and (9)), averages the duals (Eq. (3) and (4)), and distributes the result among all agents holding a copy of the state. In contrast to other strategies, the one proposed here allows for agents holding duplicated variables to remain agnostic to which other agents currently hold the same data, thereby simplifying the exchange of duals significantly.

## IV. Experiments

In this section, we present a comparative study of the proposed system with different degrees of collaboration while analyzing their potential trade-offs. In the presented experiments, individual agents are run as independent processes within a single machine (8-core AMD Ryzen 4700U @1.4GHz). As they operate concurrently on replayed pre-recorded sensor data, they communicate data asynchronously and on-demand as described in Section III. NetVLAD image descriptors for loop closure are precomputed offline. Note that while this communication is virtually instantaneous within the same machine, the stochasticity of the per-agent processes effectively creates delays between the message generation and its processing in different agents. The pre-recorded sensor data is replayed at 5% of the original speed to accommodate for the limited capacity of the single machine concurrently running all agents. At the same time, we limit computational capacity by enforcing at most one iteration of ADMM per second (adjusted to the playback speed) within each agent, and a maximum of 10 iterations for Eq. (5), to mimic realistic computational constraints on the agents. Parameters for ADMM are set to $\gamma = 100$ and $\eta = 0.9$ and remain fixed throughout all experiments.

### A. Evaluation on EuRoC

We reinterpret the five Machine Hall (MH) sequences from the EuRoC dataset [38] as multiple agents concurrently

sensing the same environment. The final keyframe pose estimates are evaluated against the provided ground truth with the toolbox [39] and the Root Mean Square Error (RMSE) is presented in Table I using multiple configurations of the proposed system. Here, the "single-agent" configuration refers to the custom-made single-agent pipeline (Section III-A) without collaboration among agents, serving as a baseline against other state-of-the-art single-agent SLAM system such as ORB-SLAM3 [5]. It should be highlighted, that compared to [5], the custom single-agent SLAM system does not employ advanced keyframe generation strategies or complex map-culling techniques. Moreover, as the current system fully relies on bundle adjustment for loop correction (i.e. without initial PGO-based correction as in [5]), loops containing large drift are still challenging for the proposed optimization scheme, yielding generally worse RMSEs shown here for reference.

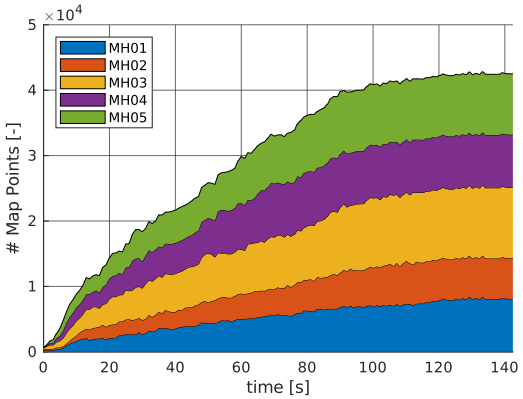
Among the collaborative setups, "no map sharing" refers to the configuration where inter-agent constraints are added when loop-closures are detected via matched map points, but no additional map data is fetched for relocalization. Conversely, the "map sharing" configuration embodies the full proposed system, where agents fully exploit covisibility information to relocalize against areas previously mapped by other agents. For these collaborative configurations we not only report the RMSE of the individually 6DoF-aligned trajectories to ground truth in Table I, but also the RMSE resulting from the combination of all the estimated trajectories, aligned using a single 6DoF transformation (i.e. 'global'). Additionally, Fig. 4 depicts the number of map points in each agent's local storage over time, i.e. a proxy for the map size as map points may be duplicated across agents.

Results in Table I indicate that the single-agent configuration outperforms the multi-agent ones, despite more information being available to all the agents. Note, however, that the iterative nature of ADMM may lead to a slower convergence rate in the number of iterations as agents can only update their individual bundle adjustment problems asynchronously. As one iteration of ADMM occurs at most once per second in this experiment, the agents may be operating on a partially converged problem as new sensor data is processed, yielding only a negligible drop in performance in simple trajectories, but producing a more notable effect on the most challenging ones (i.e. MH04, MH05).

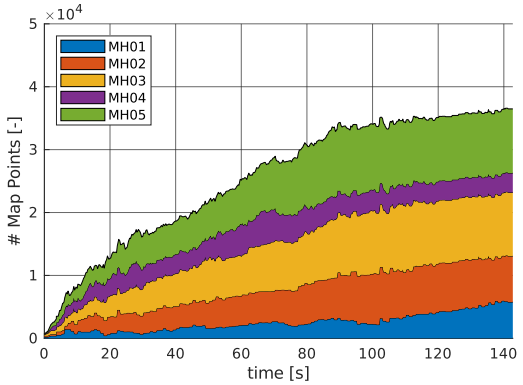
The accuracy of the proposed system using "map sharing" slightly underperforms compared to the one using the "no map sharing" configuration. In the "map sharing" configuration, agents can reuse other agents' data by dynamically transferring keyframes and map points among them, creating fluctuations as depicted Fig. 4b. This leads to an overall smaller number of keyframes and map points (here over $15\%$), and thus, an overall more weakly constrained SLAM problem (e.g., with fewer map point observations) compared to the "no map sharing" configuration. Depending on the system requirements, such a trade-off may be desirable as the noticeable keyframe and map point reductions lead to a smaller memory footprint as well as smaller optimization problems which, in turn, result in lower computational

TABLE I: RMSE in meters on MH sequences of EuRoC.

| MH Sequence | 01 | 02 | 03 | 04 | 05 | Global |
|---|---|---|---|---|---|---|
| ORB-SLAM3 [5] | 0.029 | 0.019 | 0.024 | 0.085 | 0.052 | - |
| Ours, Single-Agent | 0.044 | 0.043 | 0.068 | 0.130 | 0.166 | - |
| Ours, No Map Sharing | 0.049 | 0.054 | 0.102 | 0.306 | 0.285 | 0.235 |
| Ours, Map Sharing | 0.066 | 0.071 | 0.218 | 0.324 | 0.287 | 0.283 |




(a) No map sharing.




(b) Map sharing.

Fig. 4: Number of map points in each agent's local storage.

requirements for individual agents and better scalability (as outlined in Section IV-B).

The communication requirements for both collaborative configurations is depicted in Fig. 5, specifying the effort for map transfers (i.e. sharing of map points and keyframes), loop closures (i.e. candidate selection queries and geometric verification), and optimization (i.e. exchange of optimized states and duals). Using the "no map sharing" configuration, the map transfer is only limited to communicating merged map points upon cross-agent loop closure detections, whereas in the "map sharing" configuration additional keyframes and map points are transferred as described in Section III. As in the "no map sharing" configuration relocalization in other agents' maps is not possible, more keyframes are generated, resulting in additional loop closure queries and increasing the associated communication cost. By transferring map points and keyframes in the "map sharing" configuration, the agents also share fewer duplicate states, resulting in a smaller communication footprint for the optimization. While the total communication requirements of both collaborative configurations are similar in this experiment, despite con-

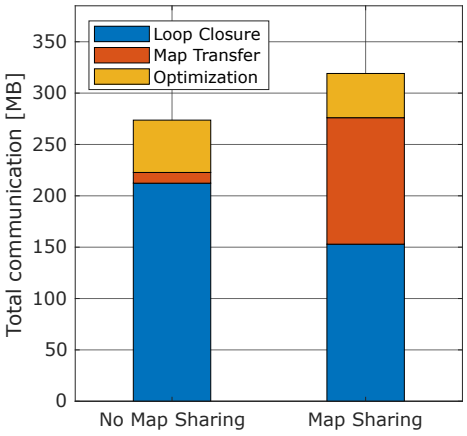



Fig. 5: Total amount of data exchanged between agents.

sidering only 5 agents and without significant map overlap, in the next experiment we explore how the "map sharing" configuration exhibit significantly better scalability in more specific scenarios.

### B. Evaluation of scalability

We evaluate the proposed approach on a self-recorded dataset featuring eight video sequences in an indoor office environment (see Fig. 1, depicting only 3 robots for clarity). These sequences have very high overlap in space and exhibit similar viewpoints, resembling a scenario that could occur, for instance, in collaborative AR / VR setups in enclosed spaces. The presented analysis focuses on the growth of computational and communication requirements as a function of an increasing number of agents.

*1) Computational cost:* The computational cost of the distributed optimization is dominated by the number of keyframes in the problem (Eq. (5)), and thus we use a total number of collectively optimized keyframes by the agents as a proxy for the overall computational cost (see Fig. 6, top). With map sharing disabled, each agent eventually maps the entire area by itself, causing the total number of keyframes to scale linearly with the number of agents. The map-sharing strategy, on the other hand, stabilizes towards a constant number of keyframes as the complete space is mapped in a distributed manner by all the agents. Sparing any other more advanced techniques, the proposed approach allows the handling of overall larger scenes, as the map and thus, the size of the optimization problems are in general more evenly distributed. Note that, while the local storage management described in Section III-B does not explicitly enforce such an even distribution among agents, it organically occurs if individual agents explore to similar degree different areas of the scene.

*2) Communication cost:* Figure 6 (bottom) shows the total data sent over the system as a function of the number of agents. Without map sharing, the communication effort scales quadratically with the number of agents. This is expected when all agents have the chance to map the whole area, as they can potentially establish loop closure to all the other agents

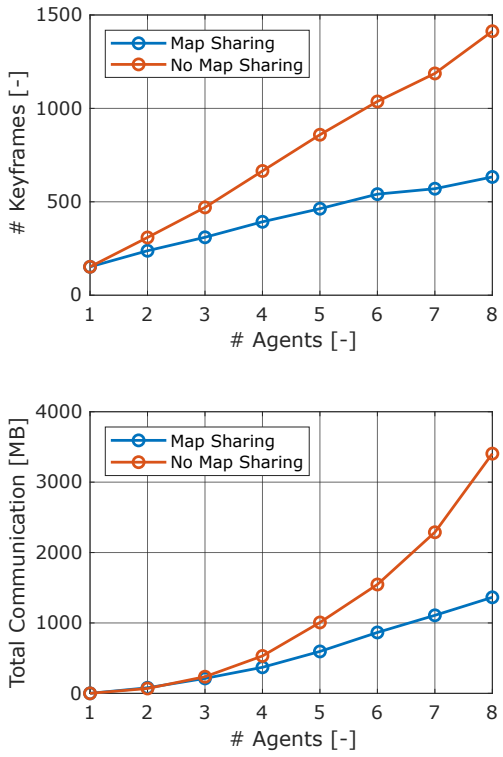



Fig. 6: Total number of keyframes produced collectively by all agents (top) and total data sent in MB (bottom) as a function of the number of agents.

for each new keyframe. Moreover, map points merged as a result of these loop closures become potentially shared among all agents, making the communication effort in the ADMM optimization also scale quadratically to keep the estimates consistent. Using map sharing, not only the number of shared states is reduced by design, lessening the communication effort related to optimization, but also, as the number of total keyframes remains proportional to the extent of the mapped scene and not the number of robots, the communication effort is also reduced.

## V. Conclusions

Acknowledging the advantages of camera relocalization in single-agent SLAM and centralized collaborative SLAM, this paper proposes a covisibility-based, peer-to-peer map sharing strategy that enables its application to decentralized collaborative SLAM systems. Driven by the need for the shared data to be consistent in the distributed system, we opt for asynchronous ADMM as the optimization backbone. The presented experimental evaluations show a significant reduction of redundancy in the map representation, as well as improvements in scalability in terms of total computational effort and communication overhead. Future work will involve the extension of our approach with strategies to handle large loop closures and addressing issues arising when dealing with delayed communication and bandwidth limitations.

## REFERENCES


[1] K. Sartipi, R. C. DuToit, C. B. Cobar, and S. I. Roumeliotis, "Decentralized visual-inertial localization and mapping on mobile devices for augmented reality," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2145–2152. 1

[2] P. Schmuck and M. Chli, "Ccm-slam: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *Journal of Field Robotics*, vol. 36, no. 4, pp. 763–781, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21854 1, 2

[3] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234. 1, 3

[4] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018. 1, 2

[5] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021. 1, 3, 5

[6] J. Morrison, D. Gálvez-López, and G. Sibley, *MOARSLAM: Multiple Operator Augmented RSLAM*, 01 2016, pp. 119–132. 1

[7] P. Schmuck, T. Ziegler, M. Karrer, J. Perraudin, and M. Chli, "Covins: Visual-inertial slam for centralized collaboration," in *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2021, pp. 171–176. 1, 2

[8] L. Riazuelo, J. Civera, and J. M. Montiel, "C2tam: A cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, 2014. 1, 2

[9] T. Ziegler, M. Karrer, P. Schmuck, and M. Chli, "Distributed formation estimation via pairwise distance measurements," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3017–3024, 2021. 1, 2, 4

[10] T. Cieslewski and D. Scaramuzza, "Efficient decentralized visual place recognition using a distributed inverted index," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 640–647, 2017. 1, 2

[11] ——, "Efficient decentralized visual place recognition from full-image descriptors," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2017, pp. 78–82. 1, 2, 3

[12] M. Giamou, K. Khosoussi, and J. P. How, "Talk resource-efficiently to me: Optimal communication planning for distributed loop closure detection," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3841–3848. 1, 2

[13] Y. Tian, K. Khosoussi, M. Giamou, J. P. How, and J. Kelly, "Near-optimal budgeted data exchange for distributed loop closure detection," *CoRR*, vol. abs/1806.00188, 2018. [Online]. Available: http://arxiv.org/abs/1806.00188 1, 2

[14] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How, "Asynchronous and parallel distributed pose graph optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5819–5826, 2020. 1, 2

[15] Z. Peng, Y. Xu, M. Yan, and W. Yin, "Arock: An algorithmic framework for asynchronous parallel coordinate updates," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A2851–A2879, 2016. [Online]. Available: https://doi.org/10.1137/15M1024950 1, 2, 4

[16] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1286–1311, 2017. [Online]. Available: https://doi.org/10.1177/0278364917732640 1, 2

[17] M. A. Nitsche, G. I. Castro, T. Pire, T. Fischer, and P. De Cristóforis, "Constrained-covisibility marginalization for efficient on-board stereo slam," in *2017 European conference on mobile robots (ECMR)*. IEEE, 2017, pp. 1–6. 1, 2

[18] A. Cunningham, M. Paluri, and F. Dellaert, "Ddf-sam: Fully distributed slam using constrained factor graphs," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3025–3030. 2

[19] A. Cunningham, V. Indelman, and F. Dellaert, "Ddf-sam 2.0: Consistent distributed smoothing and mapping," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 5220–5227. 2

[20] R. Dubois, A. Eudes, and V. Frémont, "On data sharing strategy for decentralized collaborative visual-inertial simultaneous localization and mapping," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2123–2130. 2

[21] R. Murai, J. Ortiz, S. Saeedi, P. H. Kelly, and A. J. Davison, "A robot web for distributed many-device localisation," *arXiv preprint arXiv:2202.03314*, 2022. 2

[22] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How, "Distributed certifiably correct pose-graph optimization," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2137–2156, 2021. 2

[23] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual slam," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2466–2473. 2

[24] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, "Door-slam: Distributed, online, and outlier resilient slam for robotic teams," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1656–1663, 2020. 2

[25] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems," *IEEE Transactions on Robotics*, pp. 1–17, 2022. 2

[26] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012. 2

[27] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017. 2

[28] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager, "A survey of distributed optimization methods for multi-robot systems," *arXiv preprint arXiv:2103.12840*, 2021. 2

[29] S. Choudhary, L. Carlone, H. I. Christensen, and F. Dellaert, "Exactly sparse memory efficient slam using the multi-block alternating direction method of multipliers," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1349–1356. 2

[30] A. Eriksson, J. Bastian, T.-J. Chin, and M. Isaksson, "A Consensus-Based Framework for Distributed Bundle Adjustment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[31] R. Zhang, S. Zhu, T. Fang, and L. Quan, "Distributed very large scale bundle adjustment by global camera consensus," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 29–38. 2

[32] M. Karrer and M. Chli, "Distributed variable-baseline stereo slam from two uavs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 82–88. 2, 4

[33] L. Kneip and P. Furgale, "Opengv: A unified and generalized approach to real-time calibrated geometric vision," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1–8. 2

[34] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2352–2359. 2

[35] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307. 3

[36] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, 1976. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0898122176900031 4

[37] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015. 4

[38] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. 4

[39] M. Grupp, "evo: Python package for the evaluation of odometry and SLAM." https://github.com/MichaelGrupp/evo, 2017. 5